I. P. Sharp Associates

The SHARP APL Utility Library

THE SHARP APL UTILITY LIBRARY

Version 1
1 December 1983

Printed in Canada
January 1984
Publication Code: 0233 8401 E1

Contents

1 Introduction 1

2 Fields Associated With a Utility
- Field 1: Name 5
- Field 2: Version 5
- Field 3: Timestamp 5
- Field 4: Summary 5
- Field 5: Changes 5
- Field 6: Description 6
- Field 7: Details 6
- Field 8: Categories 6
- Field 9: Reviews 6
- Field 10: Deleted 6
- Field 11: Secondary 6
- Field 12: Type 7
- Field 13: Definition 7
- Field 14: Subutilities 7
- Field 15: References 7

3 Retrieving A Utility From The Utility Library
- Title Expressions 8
- *UCATINDEX* 8
- *UCATDESCRIBE* 9
- *UCATFIND* 10
- *UINCAT* 10
- *USUMMARY* 10
- *UFIND* 10
- *UDESCRIBE* 11
- *UDEFINE* 12
- *UDEFNAMES* 13
- *UTITLES* 13
- *UNC* 13
- *URETRIEVE* 13
- *UNION*, *INTERSECT*, and *MINUS* 13
- *UCATALOGUE* 14
- *UREVIEW* 15

4 Submitting A Utility To The Utility Library
- *UEDIT* 16
- *UVALIDATE* 17
- *USUBMIT* 17
- *URESUBMIT* 18
- *UREVISE* 18
- *UDELETE* 18

Appendix: The Category Index 19

Chapter 1

INTRODUCTION

Most programmers spend a large amount of time writing programs that other people have written, are writing, or will someday write. It has been widely recognized that at least part of the solution to this problem lies in providing a repository of "utilities"--the programs commonly used by programmers in writing systems. At the First Technical Meeting of I.P. Sharp Associates, a working group was formed to consider such a repository. A consensus as to its general nature was reached. Utilities were to be freely submittable by every member of I.P. Sharp Associates; they were to be categorized by the submitter according to a predefined (but extendible) set of categories; each utility was to have a detailed, easily obtainable description; there were to be no restrictions on the subutilities a utility could call. This repository has now been constructed--The SHARP APL Utility Library. Workspace 502 *ULIBRARY* contains the library access functions.

The Utility Library has been designed to make retrieval of utilities easy, even when the names of the utilities sought are unknown. For example, suppose you are writing a multi-user system with several files, and you need functions that manipulate several access matrices at a time. The first thing to do is look over the index of Utility Library categories. If you do not have a copy of the index nearby, you can obtain one as follows.

```
      UCATINDEX
```

The index is about a page long. The Appendix lists the index as of 27 August 1983.

Looking over the index you find category 9--Files. Category 9 has several subcategories, in particular, category 32--File Permission Control--and category 11--Concurrent Use Of Files (file sharing). If the utilities you want are in the library, they will be found in category 32 or 11, likely both. Employing <*UINCAT*>, the title (Name and Version) of each utility in categories 32 and 11 can be obtained as follows. (Note: utilities shown in this manual are examples only, and do not necessarily exist in the Utility Library.)

```
      UINCAT 32 11
AWE/FRDAC.1
AWE/FSTAC.1
FILEAC/GIVEACCESS.1
FILEAC/REMOVEACCESS.2
FILEAID/ACMAT.1
FJP/FILES/FHOLD.1
FJP/FILES/FSTIE.1
.
.
.
```

Notice there is a second version of *FILEAC/REMOVEACCESS*, but not a first version. The first version must have been deleted.

The utilities you seek can now be easily found.

```
      USUMMARY UINCAT 32 11
AWE/FRDAC.1              SIMILAR TO ⎕RDAC; RETURNS 0 3⍴0 IF NO ⎕RDAC ACCESS.
AWE/FSTAC.1              SIMILAR TO ⎕STAC; RETURNS 1 IF SET WAS MADE, ELSE 0.
```

```
FILEAC/GIVEACCESS.1      SETS AC MATS FOR SPECIFIED FILES; OR'S EXISTING ACCTS.
FILEAC/REMOVEACCESS.2    REMOVES SPECIFIED ACCESS, FOR SELECTED USERS, FROM
                         SPECIFIED FILES.
FILEAID/ACMAT.1          AN AID IN FORMING FILE AC MATRICES.
.
.
.
```

<USUMMARY> returns a short summary of each utility specified in its argument.

It seems certain the utilities wanted are *FILEAC/GIVEACCESS* and *FILEAC/REMOVEACCESS*. As well, *FILEAID/ACMAT* looks as though it might be useful. To get more details, execute the following.

```
      UDESCRIBE 'FILEAC/(GIVEACCESS.1 REMOVEACCESS.2) FILEAID/ACMAT.1'

TITLE:        FILEAC/GIVEACCESS.1

TYPE:         FUNCTION

SUMMARY:      SETS AC MATS FOR LISTED FILES; OR'S EXISTING ACCTS.

TIMESTAMP:    1982-10-14 05:22:36.378

DESCRIPTION:
FILEREF GIVEACCESS ACCESSMATRIX

SETS THE ACCESS MATRICES FOR THE REFERENCED FILES; OR'S THEM IF AN ENTRY FOR
AN ACCOUNT ALREADY EXISTS.

<FILEREF> MAY BE ONE OF THE FOLLOWING.

        * NUMERIC VECTOR OF FILE TIE NUMBERS
        * TWO COLUMN MATRIX, WITH THE FIRST COLUMN CONTAINING TIE NUMBERS
          AND THE SECOND CONTAINING PASSNUMBERS
        * CHARACTER VECTOR OF FILE NAMES, DELIMITED BY COMMAS
        * MATRIX NAMELIST OF FILE NAMES

<ACCESSMATRIX> SHOULD BE A 2 OR 3 COLUMN MATRIX CONTAINING ACCOUNT NUMBERS,
ACCESS NUMBERS, AND (OPTIONALLY) PASSNUMBERS.  IF <ACCESSMATRIX> HAS ONLY TWO
COLUMNS, THE PASSNUMBERS ARE LEFT UNCHANGED OR ASSUMED TO BE 0.

<GIVEACCESS> IS ⎕IO-INDEPENDENT.  IT HAS BEEN THOROUGHLY TESTED.

REVIEWS:
[1982-12-30 19:22:33.860]

UNLIKE THE SYSTEM FUNCTIONS ⎕EX, ⎕NC, ⎕PACK, ETC., A MATRIX NAMELIST ARGUMENT
REQUIRES THE NAMES BE LEFT-JUSTIFIED ON EACH ROW.

                                    JOHN ADAMS

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
TITLE:      FILEAC/REMOVEACCESS.2

TYPE:       FUNCTION

SUMMARY:    REMOVES SPECIFIED ACCESS, FOR SELECTED USERS, FROM SPECIFIED
            FILES.

.
.
.
```

Clearly, *FILEAC/GIVEACCESS* and *FILEAC/REMOVEACCESS* are the required functions. As well, *FILEAID/ACMAT* will be useful. So, define the utilities in your system.

```
      )LOAD MULTIUSERWS
      )FNS
ALTER   GETDATA GIVEACCESS      JUSTIFY . . .
      )COPY 502 ULIBRARY UDEFINE
      UDEFINE 'FILEAC/(GIVEACCESS REMOVEACCESS) FILEAID/ACMAT'
 GIVEACCESSΔA
 REMOVEACCESS
 ACMAT
  PASSNO
   KEYS
```

If you do not specify a version number, the latest is assumed.

The result of <*UDEFINE*> is a character matrix containing the names under which the utilities are now defined in the workspace. Here, it indicates *FILEAC/GIVEACCESS* is defined under *"GIVEACCESSΔA"*; a new name had to be used because a different (visible) object named *"GIVEACCESS"* already existed. It also indicates utilities <*PASSNO*> and <*KEYS*> are defined. <*PASSNO*> is a subutility of *FILEAID/ACMAT*; <*KEYS*> is a subutility of <*PASSNO*>. As with all subutilities, they are automatically defined by <*UDEFINE*>.

If a visible object named *"PASSNO"* had already existed (with a different definition than the subutility), then the subutility would have been defined under *"PASSNOΔA"* and all references to it, in <*ACMAT*>, would have been changed to *"PASSNOΔA"*.

As illustrated in the example, searching for a utility whose name is unknown usually requires three steps:

* determining the categories in which it might be,
* reading summaries of the utilities in these categories, and
* reading selected descriptions, such as those printed by <*UDESCRIBE*>.

Retrieval can always be accomplished by these three steps. The example showed how to do them on-line, that is, how to search while signed on to SHARP APL. You can also search off-line, using the Utility Library Catalogue. The catalogue consists of three parts, one for each step:

* the category index,
* summaries of the utilities in each category, and
* detailed descriptions of each utility in the library.

The catalogue can be *HSPRINT*'d by executing '*UCATALOGUE* 1'.

As is clear from the behaviour of <*UDESCRIBE*>, each utility has associated with it a number of different fields, which contain various facts. These fields are discussed individually in the next chapter, Fields Associated With A Utility. The remaining two chapters, Retrieving A Utility From The Utility Library and Submitting A Utility To The Utility Library, discuss the two main uses of the Utility Library: retrieval and storage of utilities. For most people, the former use, utility retrieval, will be the main one. It should not be the only one. The library would be valueless if it contained no utilities, and utilities only get in the library when people submit them. So be sure to contribute your share.

NOTE Since this manual was written, a new function, <*USEARCH*>, has been included with the Utility Library. <*USEARCH*> is a simple, conversational function designed to guide you in searching for a utility. It can be used even if you do not have an understanding of the functions described above. To use it, load workspace 502 *ULIBRARY* and enter *"USEARCH"*.

Chapter 2

FIELDS ASSOCIATED WITH A UTILITY

Associated with each utility in the Utility Library are a number of different "fields". The Version of a utility, its Description, and its Definition are examples of fields. Fields are both named and numbered. The field named "Description", for example, is field number 6. Field numbers are stored in appropriately named variables. For example, 6 is stored in the variable *<DESCRIPTION>*. These variables are grouped under *"FIELDS"*.

In this chapter, each of the fields is described. The fields fall into two sets. The first set consists of the fields used (explicitly) when searching for a utility, for example, the Summary and Description fields. The first nine fields fall into this set. The remaining six fields serve other purposes. For example, the Subutilities field of a utility lists the titles of all its required subutilities. The descriptions of these six fields can be skimmed.

Field 1: Name

A utility's Name consists of one or more APL names separated by slashes, as in *"RCM/TERMFNS/AJ860/HTAB∆SET"*. In general, the first APL name of a utility's Name will be the author's mailbox code; the last APL name will be the name the utility will have when defined in a workspace. The last APL name is known as the "short name" of a utility.

Field 2: Version

The Utility Library may contain different versions of a utility, much as a library of books may contain different editions of a certain book. Different versions of a utility are distinguished by their Version field. The Version field of the first version of a utility is 1, the Version field of the second version of a utility is 2, etc.

The "title" of a utility is formed by catenating its Name (field 1) with a dot and its Version. For example, *"ROHAN/QSAVE.2"* is the title of the second version of *ROHAN/QSAVE*. Titles are necessarily distinct.

Field 3: Timestamp

This field specifies the time the utility was originally submitted to the Utility Library, in *□TS*-format.

Field 4: Summary

The Summary is a succinct description of the utility, giving a general overview of what the utility does or what it might be used to do. It is a single line, at most 100 characters in length.

Field 5: Changes

This field lists the changes from other versions of the utility, and describes why the changes were made. Its maximum width is 80 characters. It is usually empty if the utility's version number is 1.

Field 6: Description

This field contains a fairly detailed description of the utility's behaviour, and suggests how the utility might be used. It will usually indicate whether the utility is restartable, origin independent, etc., and frequently contains several examples. As well, it might contain an indication of how thoroughly the utility has been tested, where the utility originated, and any important implementation details. The field is limited to a maximum of 50 lines and 2000 characters. Its maximum width is 80 characters.

Field 7: Details

This field is usually empty. If it is not empty, it will contain information about the utility that could not conveniently be fit in the Description field (field 6). It might, for example, contain test runs of the utility to support claims of correctness or speed, or describe how the algorithm used was derived. There is no limit on the length of this field.

Field 8: Categories

The Categories field is a vector of positive integers specifying which categories the utility is in. Execute *<UCATINDEX>* to see the current index of categories.

Field 9: Reviews

This field contains reviews of the utility. It is entered by the users of the utility, not the submitter, and is thus different in an important way from all other fields. Everyone who uses a utility is encouraged to submit a review of it. Reviews can be submitted via *<UREVIEW>*. They may contain comments of any sort related to the utility.

The remaining six fields are not normally used explicitly when searching for a utility. Their descriptions may be elided.

Field 10: Deleted

Any utility may be deleted from the Utility Library by its submitter. This may be done at any time and for any reason. A deleted utility remains in the library for those who depend on it; instead of being removed from the library, a deleted utility is marked "deleted". The Utility Library searching functions will ignore a deleted utility, so people who do not know of its existence will never encounter it.

A utility is marked as "deleted" by having its Deleted field be non-empty. Normally, a non-empty Deleted field will be a character vector detailing why the utility was deleted.

Field 11: Secondary

This field is `1` if and only if the sole purpose of the utility is to be used by or with other utilities, that is, as a subutility. If the utility is to be used by or with other utilities, but might also be useful as a utility in its own right, this field will be `0`. The Utility Library searching functions will ignore a secondary utility, so people who do not know of its existence will never encounter it.

Field 12: Type

This field is `1` if the utility is a function and `0` if the utility is an array. Note that packages are not (currently) permitted to be utilities.

Field 13: Definition

If the utility is a function, this field contains its canonical representation (*⎕CR*). If the utility is an array, this field contains its value.

Field 14: Subutilities

A "subutility" is a utility that is to be defined, by *<UDEFINE>*, whenever some other utility is. Usually, a subutility satisfies one of the following: another utility might use (call) it, or another utility might be used with it. An example of the latter occurs in *PLOT 2 POLYFIT COORDS*. Here, *<POLYFIT>* finds the best degree-2 polynomial for the given *<COORDS>*; *<PLOT>* plots the result. If it is considered desirable to have *<POLYFIT>* available whenever *<PLOT>* is, so that *<PLOT>* can always be used with *<POLYFIT>*, then *<POLYFIT>* will be a subutility of *<PLOT>*. Note that this occurs even though *<POLYFIT>* may be a useful utility in its own right.

The Subutilities field is a character matrix with one title (Name and Version) per row. Each title designates a (sub)utility that will automatically be defined, in the workspace, whenever the utility is. Any utility in the Utility Library may serve as a subutility of any other.

Field 15: References

If the utility is an array, this field is empty. Otherwise, it contains information that may be necessary to define the subutilities in the workspace. The field exists because a utility may not always be defined in the workspace with the same name with which it was submitted to the Utility Library. This name changing occurs only if there are name conflicts. See the description of *<UDEFINE>* for more details.

The References field is a three-column matrix specifying the location of each reference (within the canonical representation) to a subutility. The first column is a 0-origin index into the Subutilities field (field 14); each element of the first column thus determines a subutility being referenced. The second and third columns are 0-origin row and column indices to the location of a reference within the canonical representation.

Note that the existence of this field implies the short (APL) name of a subutility should appear explicitly in the canonical representation each time the subutility is referenced. For example, if utility *<FN>* calls utility *<SUBFN>*, then *<FN>* should not contain an expression such as `⍎'SUB','FN'`, since this expression references *<SUBFN>*, but the string `'SUBFN'` does not appear within it. *<FN>* may, however, contain *RESULT←(⍳7) SUBFN ARGUMENT*, `⍎CONDITION/'Z←SUBFN ARG'`, and `⍎'⍎COND/''SUBFN ¯1'''`, since the string `'SUBFN'` appears explicitly in their character representations.

Chatper 3

RETRIEVING A UTILITY FROM THE UTILITY LIBRARY

The Introduction described one simple method of retrieving utilities from the Utility Library. Most of the time that method will be satisfactory. Occasionally, however, more sophisticated methods may be desired. This chapter describes all the retrieval functions of the Utility Library, including those illustrated in the Introduction.

Title Expressions

Many Utility Library functions take, as one of their arguments, a list of titles (Names and Versions) of utilities. Such arguments are named *"UTILITIES"* in a function header, and must have one of the following two formats: a matrix with one left-justified title per row or a "title expression". Examples of title expressions are given below.

RBE/ENCLOSE.1	Version 1 of *RBE/ENCLOSE*.
RCM/TERMINALS/INPUT/ARBIN.3	Version 3 of *RCM/TERMINALS/INPUT/ARBIN*.
DBA/VTOM.(1 5)	Versions 1 and 5 of *DBA/VTOM*.
DBA/VTOM	If no version is specified, the latest version is chosen. This refers to the latest version of *DBA/VTOM*.
GBO/FILES/(FREAD.2 *FRESIZE)*	Version 2 of *GBO/FILES/FREAD* and the latest version of *GBO/FILES/FRESIZE*.
SLC/FNS/(FNEDIT.1 *FNSHOW*.1)	Version 1 of *SLC/FNS/FNEDIT* and *SLC/FNS/FNSHOW*.
SLC/FNS/(FNEDIT FNSHOW).1	As above.
SLC/(FNS/FNEDIT FNS/FNSHOW).1	As above.
*GOTT/TERMFNS/(OUT/(TABS/AJ*832 *ARBOUT*.(1 3)) *IN/ARBIN*.2)	The latest version of *GOTT/TERMFNS/OUT/TABS/AJ*832, versions 1 and 3 of *GOTT/TERMFNS/OUT/ARBOUT*, and version 2 of *GOTT/TERMFNS/IN/ARBIN*.

Parentheses may be nested to any depth. A right parenthesis may not be followed by a slash.

Stars may appear in title expressions. They can be used to match the Name or Version of any "active" (non-deleted and non-secondary) utility, as illustrated below.

PRA/ASK.*	All active versions of *PRA/ASK*.
PRA/.**	All active versions of utilities whose titles begin with *"PRA/"*. This includes, for example, *PRA/FILES/OUTPUT/FAPPEND*.2.
*PRA/**	All active latest versions of utilities whose titles begin with *"PRA/"*.
DWBY/(INPUT/ OUTPUT/TABS/*.1)*	All active latest versions of utilities whose titles begin with *"DWBY/INPUT/"*, together with all active first versions of utilities whose titles begin with *"DWBY/OUTPUT/TABS/"*.

A star may not be followed by a slash.

A circle used in place of a star permits matching of non-active utilities submitted by *⎕AI[⎕IO]*. This allows you to determine the titles of all of your utilities, both active and inactive.

```
FJP/o.o          ⍝ If you are "FJP", this refers to all of your utilities,
                 ⍝ both active and inactive. Otherwise, this is the same
                 ⍝ as "FJP/*.*".
FJP/o            ⍝ The latest versions of the above.
```

INDEX←UCATINDEX

This function returns an index of the Utility Library categories. The index is about a page long. It consists of a revision date and the number and description of each category. Some descriptions are indented, indicating that the corresponding category is a subcategory of some category indexed above it. Utilities in a subcategory are in all categories superior to that subcategory.

```
     UCATINDEX
REVISION DATE: 1983-08-27

 1  MISCELLANEOUS
 2  STRUCTURAL TRANSFORMATIONS  ⍝ RESHAPING, CATENATING, TRANSPOSING, ETC.
40  SELECTING FROM ARRAYS  ⍝ INDEXING, TAKE, COMPRESSION, ETC.
41  BOOLEAN ARRAYS
 .
 .
 .
20  TERMINAL INPUT/OUTPUT
21     TERMINAL INPUT
22     TERMINAL CONTROL
23        TABS
45  GRAPHICS AND PLOTTING
24  EDITING
25  FORMATTING
30     REPORT FORMATTING
 .
 .
 .
```

Note that the category numbers are not in order. This is because when a new category is added, the Utility Librarian indexes it near related categories. For example, when the Graphics and Plotting category was added, the librarian indexed it near Terminal Input/Output. The new category was numbered 45 because categories 1 through 44 previously existed.

Categories may occasionally be eliminated. When this occurs, other categories will not be renumbered. For example, if category 38 is ever eliminated, category 39 will not be renumbered 38. Rather, there will not be a thirty-eighth category.

Suggestions for expanding the list of categories or improving the index are always welcome. They should be sent to the Utility Librarian, whose mailbox code is *"ULIB"*.

DESCRIPTIONS←UCATDESCRIBE CATNUMS

This function returns descriptions of the categories specified by *<CATNUMS>*. *<DESCRIPTIONS>* is a character matrix with one description per row. Each description is the same as the corresponding description in the category index.

```
      UCATDESCRIBE 21 11
21  TERMINAL INPUT
11  CONCURRENT USE OF FILES  ⍝ FILE SHARING
```

CATNUMS←UCATFIND STRING

The result is a vector of the numbers of categories associated with *STRING*. For example, *UCATFIND 'FILES'* comprises the numbers of all categories having anything to do with files. *UCATFIND ''* comprises all valid category numbers.

TITLES←UINCAT CATNUMS

TITLES is a character matrix containing the titles of all active utilities that are in any of the specified categories. For example, *UINCAT* 21 11 contains the titles of all utilities in either category 21 or category 11. Each row of *TITLES* holds one left-justified title.

SUMMARIES←USUMMARY UTILITIES

The result is a vector of enclosures, with one enclosure for each of the specified utilities. Each enclosure contains the title (Name and Version) and Summary (field 4) of a utility, formatted to width ⎕*PW*, as illustrated below.

```
      ⎕PW←78
      ⍴SUMS←USUMMARY 'MBOXCODE/* JSL/(TEST/CPUTIME ADRS.2)'
4
      SUMS
MBOXCODE/SHORTNAME.1     A UTILITY.
MBOXCODE/GROUP/FN.3      ANOTHER UTILITY.
JSL/TEST/CPUTIME.1       COMPUTES THE APPROXIMATE MILLI-CPU USED FOR N
                         REPETITIONS OF A LINE OF CODE.
JSL/ADRS.2               A DEPARTMENTAL REPORTING SYSTEM.
```

TITLES←[UTILITIES] UFIND STRING IN FIELDNUM

(*<IN>* is a supporting function in workspace 502 *ULIBRARY*, which encloses and catenates its arguments.) *TITLES* is a left-justified character matrix holding the titles of all the specified utilities that contain *STRING* in field *FIELDNUM*. If *<UTILITIES>* is not specified, the searching is done over all (active) utilities in the Utility Library. For example,

```
      UFIND 'COMPLEX' IN DESCRIPTION
```

comprises the titles of all active utilities that contain *'COMPLEX'* in their Description field (field 6).

An important use of *<UFIND>* is determining recently-submitted utilities. For example, to find out what new utilities have been submitted during the current month, use the following.

```
      UFIND (2↑⎕TS) IN TIMESTAMP
```

<*UFIND*> is also useful when searching for a utility whose Name is only partially remembered. For example, suppose you know there is a utility that facilitates easy B-task submission. You remember the (short) name was *"SUB∆B"* or *"SUBMIT∆B"* or *"BTASK∆SUB"*--or something like that--but can recall nothing else.

```
      (UINCAT UCATFIND 'BTASKS') UFIND 'SUB' IN NAME
```

[*FIELDNUMS*] *UDESCRIBE UTILITIES*

This function prints a formatted display for fields of the specified utilities.<*FIELDNUMS*> designates the fields to be obtained.Its default value is *DELETED*, *SECONDARY*, *TYPE*, *SUMMARY*, *TIMESTAMP*, *CHANGES*, *DESCRIPTION*, *REVIEWS*.(The fields not included are Name, Version, Details, Categories, Definition, Subutilities, and References. Name and Version are not included because utility titles are always printed.)

```
      (DESCRIPTION,CHANGES,REVIEWS) UDESCRIBE 'MBCODE/*.*'

TITLE:       MBCODE/COST.1

DESCRIPTION:
PRINTS A TABLE SIMILAR TO THAT DISPLAYED AT SIGNOFF, ONLY SHOWING ACTUAL
COSTS (IN U.S. DOLLARS).

<COST> HAS BEEN THOROUGHLY TESTED.  IT WORKS WHETHER ⎕IO IS 0 OR 1.

REVIEWS:
[1983-01-31 06:17:35.091]

THE RESULT IS DISPLAYED, RATHER THAN RETURNED.  MOREOVER, ⎕FMT IS USED, WHICH
MAKES IT A NUISANCE TO USE WITH ⎕OUT AND SLIGHTLY MORE EXPENSIVE THAN IT
NEEDS TO BE.                                        --DOUGLAS J. KEENAN

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


TITLE:       MBCODE/EDIT.2

DESCRIPTION:
NEWTEXT←[COMMANDS] EDIT OLDTEXT

<EDIT> IS A STRING-ORIENTED TEXT EDITOR.  IT CAN BE USED TO INTERACTIVELY
EDIT A CHARACTER OR NUMERIC ARRAY OF ANY SHAPE.  A DISTINCTIVE FEATURE OF
<EDIT> IS THAT IT HAS ONLY ONE MODE, COMMAND MODE.  IN PARTICULAR, THERE IS
NO INPUT MODE: ALL TEXT MUST BE INPUT AS THE ARGUMENT TO A COMMAND.  THOUGH
SOMETIMES SLIGHTLY CUMBERSOME, THIS LACK OF MODES CAN MAKE LIFE SIMPLER,
PARTICULARLY FOR 'NAIVE' USERS.  COMMAND MODE IS VERY SIMILAR TO THE ⎕-INPUT
MODE OF APL: ALL <EDIT> COMMANDS ARE REALLY APL FUNCTIONS, LOCALLY DEFINED BY
<EDIT>--ANY COMMANDS YOU ENTER ARE REALLY JUST CALLS TO THESE LOCAL
FUNCTIONS.  THIS MAKES COMMAND MODE FAIRLY TRANSPARENT, AND ALSO MAKES IT
REASONABLY SIMPLE TO ADD NEW COMMANDS OR CUSTOMIZE EXISTING ONES.
```

```
FOR A COMPLETE DESCRIPTION OF <EDIT>, SEE THE DETAILS FIELD.

*** EXECUTE   DETAILS UDESCRIBE 'MBCODE/EDIT.2'   FOR MORE INFORMATION

CHANGES:
SOME SUPPORT FOR FULL-SCREEN INPUT ADDED.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


TITLE:       MBCODE/SPACE.1

DESCRIPTION:
BYTES←SPACE STATEMENT

DETERMINES AN UPPER BOUND ON THE NUMBER OF BYTES REQUIRED TO EXECUTE
<STATEMENT>.  THE UPPER BOUND IS WITHIN 1 KBYTE OF THE ACTUAL NUMBER OF
BYTES REQUIRED.

<SPACE> SETS ⎕OUT TO  0 0  SO THAT ANY OUTPUT GENERATED BY <STATEMENT> WILL
BE DISCARDED.  ⎕OUT IS RESET TO ITS ORIGINAL VALUE AFTER <BYTES> HAS BEEN
DETERMINED, BUT WATCH OUT: IF <STATEMENT> EVOKES AN ERROR, <SPACE> WILL
SUSPEND WITH ⎕OUT STILL BEING  0 0.

<SPACE> HAS BEEN WELL-TESTED.  IT WILL WORK ONLY IN ORIGIN 1.
```

The line beginning with "*** is not part of the Description field of *MBCODE/EDIT*.2, but was generated by <*UDESCRIBE*>. Such lines are generated by <*UDESCRIBE*> whenever (*DESCRIPTION*, *DETAILS*) ∈ *FIELDNUMS* ↔ 1 0 and the Details field is non-empty. Note that empty fields are never displayed.

OBJNAMES←UDEFINE UTILITIES

<*UDEFINE*> defines the specified utilities and all required subutilities, in the workspace. The result, *OBJNAMES*, is a character matrix. 0 1↓*OBJNAMES* is a matrix namelist containing the (APL) names under which the utilities were defined. The names of subutilities are located immediately below the names of the utilities they support. Names of subutilities are indented one space (in 0 1↓*OBJNAMES*), names of subutilities of subutilities are indented two spaces, etc., as illustrated below.

```
      ⍴MAT←UDEFINE 'AWE/(INDEXOF.1 VTOM) MBC/UTIL SAUCE/DISTINCT'
7 11
      MAT
 INDEXOF
*VTOM          ⍝ AWE/VTOM already existed in the workspace.
 UTIL
  PSORT        ⍝ MBC/UTIL calls JIC/PSORT.
   PGRADEUP    ⍝ JIC/PSORT calls SAUCE/PGRADEUP.
  FOO∆A        ⍝ MBC/UTIL also calls MBC/FOO; see below.
 DISTINCT∆A    ⍝ Another object named "DISTINCT" existed in the workspace.
```

The first element of each row is either a space or a star. If it is a star, the utility named on that row already existed in the workspace (with the same visible definition). Note that executing '⎕*EX OBJNAMES*' will thus erase only those objects that did not previously (visibly) exist.

If a visible object already exists under the (short) name of a utility that is to be defined, the utility is renamed and all calling utilities have their references to it changed appropriately. In the above example, *MBC/UTIL* called *MBC/FOO*, and an object named *"FOO"* already (visibly) existed. So *UDEFINE* '*MBC/UTIL*' caused *MBC/FOO* to be defined under the name *"FOO∆A"*, and all references to <*FOO*>, in <*UTIL*>, were changed to references to <*FOO∆A*>. If <*FOO∆A*> had already existed, <*FOO∆B*> would have been used, etc.

All of the specified utilities must exist in order for any of them to be defined. In the above example, if *MBC/UTIL* had not existed, an error would have been evoked at <*UDEFINE*>, and no utilities would have been defined.

OBJNAMES←UDEFNAMES UTILITIES

The result of <*UDEFNAMES*> is identical to the result of <*UDEFINE*>. <*UDEFNAMES*> does not, however, define any utilities in the workspace.

TITLES←UTITLES UTILITIES

<*UTITLES*> returns a matrix containing the titles of each of the specified utilities. Each row holds one left-justified title. <*UTITLES*> is mainly used for explicitly evaluating title expressions.

CLASSNUMS←UNC UTILITIES

<*UNC*> is similar to <⎕*NC*>. Its result is a vector of integers, with ⍴*UNC UTILITIES* ↔ 1↑⍴*UTITLES UTILITIES*. The integers have the meanings indicated below.

0 The utility does not exist.	3 The utility is a function.
2 The utility is an array.	4 Ill-formed title.

FIELDS←[FIELDNUMS] URETRIEVE UTILITIES

This function is used to obtain the fields indicated by <*FIELDNUMS*> for each of the specified utilities. If <*FIELDNUMS*> is not specified, all fields of the specified utilities are retrieved. The result is a matrix of enclosures, with one row per utility, and one column per field. Each enclosure contains one field.

UNION, *INTERSECT*, and *MINUS*

Three functions for manipulating lists of utility titles (Names and Versions) are provided: <*UNION*>, <*INTERSECT*>, and <*MINUS*>. Their left and right arguments should be matrices containing one left-justified title per row. Their behaviour is illustrated below.

```
      ⎕←C←UFIND 'CONTINUOUS' IN DESCRIPTION
EEM/FIT.2
WGR/FS∆CONT.1
      ⎕←W←UTITLES 'WGR/*'
```

```
WGR/FSΔCONT.1
WGR/FSΔDISCRETE.1

      C UNION W
EEM/FIT.2
WGR/FSΔCONT.1
WGR/FSΔDISCRETE.1
      C INTERSECT W
WGR/FSΔCONT.1
      C MINUS W
EEM/FIT.2
```

The functions are often useful in restricting a library search. For example, suppose you seek a function editor. If it exists in the Utility Library, it should be in both category 24--Editing--and category 28-- (utilities dealing with) Defined Functions. So, you should proceed as follows.

```
      USUMMARY (UINCAT 24) INTERSECT UINCAT 28
```

(*UINCAT* 24 28 would return the titles of all utilities in either category 24 or category 28.) The Summaries indicate all utilities in the intersection do indeed perform function editing. Some of the utilities, however, are designed to edit functions in a package or a file, which is not what you want. Thus, the last step of your search should be to get full descriptions of each utility not in category 8--Packages--or category 9--Files. This may be done as follows.

```
      UDESCRIBE ((UINCAT 24) INTERSECT UINCAT 28) MINUS UINCAT 8 9
```

You can then read the descriptions, and choose the utility that best suits your needs.

Note that the category numbers used in this example could have been found either by using *<UCATFIND>* or by looking through the category index.

HSPRESULT←[HSPINSTRUCTS] UCATALOGUE DATE

Most library searching functions are designed to be used on-line; that is, in order to find a utility, or to see what utilities are in the library, you have to sign on to the SHARP APL system and execute functions such as *<UINCAT>*, *<UDESCRIBE>*, etc. Sometimes, however, it is more convenient to have an off-line (hard copy) list of utilities, which can be inspected as the need arises. For this reason, the Utility Library Catalogue is provided. The catalogue consists of three parts:

* the category index,
* summaries of the utilities in each category, and
* detailed descriptions of each utility in the library.

Copies of the catalogue can be obtained by executing *'UCATALOGUE 1'*. The catalogue is fairly large, and so is normally printed via *HSPRINT*. *<UCATALOGUE>* will prompt for *HSPRINT* instructions before submitting the catalogue to be printed.

Since the contents of the Utility Library are always changing, the catalogue tends to become dated quickly. Generally, however, the number of pages that change during, say, one month will be small. It is thus usually preferable to obtain replacements for obsolete pages on a regular basis, rather than getting a whole new catalogue once or twice a year. *<UCATALOGUE>* can be used to do this. Its right argument is normally a date, and the function prints replacements for any pages that

have become obsolete since that date. (*UCATALOGUE* `1` thus prints replacements for any pages that have changed since year 1--the entire catalogue.)

If the date given is recent enough, there will only be a few replacement pages. For this reason, output may be directed to the terminal instead of being *HSPRINT*'d. If called monadically, <*UCATALOGUE*> asks where output is to be directed. If called dyadically, the left argument must be *HSPRINT* instructions--print specifications followed by a jot followed by delivery instructions. A left argument will cause <*UCATALOGUE*> to act non-conversationally in submitting the request to *HSPRINT*.

The explicit result of <*UCATALOGUE*> is `''` if output was to the terminal, and the result of <*HSP*> if output was to *HSPRINT*.

MESSAGE←[*REVIEW*] *UREVIEW UTILITY*

This function is used to submit (or resubmit or delete) a review of the specified utility. <*UTILITY*> should specify the utility you wish to review. If the left argument is not specified, <*UREVIEW*> operates conversationally--you will be prompted to submit a new review, resubmit a previous review, or delete a previous review.

The optional left argument permits non-interactive reviewing. It may contain either one or two enclosures. If one enclosure, the enclosure must hold a character vector containing the (preferably signed) review to be submitted. If two enclosures, the first enclosure must hold a timestamp (in ⎕*TS*-format) specifying the review to be resubmitted or deleted; the second enclosure must hold a character vector--the review to be resubmitted, or `''` if the review is to be deleted.

The result of <*UREVIEW*> is a message indicating that the review has been successfully submitted. An error will be evoked if it could not be.

Chapter 4

SUBMITTING A UTILITY TO THE UTILITY LIBRARY

Submitting a utility to the Utility Library requires the specification of the fields associated with a utility. This is both fast and easy. In general, the only field that will take more than a minute to prepare is the Description field (field 6).

Preparation of a utility's fields is usually accomplished via the conversational function *<UEDIT>*. After the fields have been prepared, *<USUBMIT>* is used to submit them to the library. These two functions, together with other related functions, are described in this chapter.

FIELDS←UEDIT FIELDS

This is the main function used for preparing a utility to be submitted to the Utility Library. It is not used for actually submitting a utility to the library; that is done by *<USUBMIT>*. It is a conversational aid used for preparing the fields that should be completed before a utility is submitted. Its result is a vector of enclosures--the enclosed fields--to be passed to *<USUBMIT>*.

<UEDIT> proceeds by asking you to input, in turn, each of the required fields. For example, to submit *<VTOM>* to the library, you might execute the following.

```
      VTOMFIELDS←UEDIT ''
                .                  ⍝ Begin entering fields.
                .
                .
                .                  ⍝ Stop entering fields.
RESULT OF <UEDIT> ALSO SAVED IN VARIABLE <UINPUT>.
```

You should be sure to save the result, as was done above with *<VTOMFIELDS>*. Just in case you forget, the result is also saved in a global variable *<UINPUT>*; a message is displayed to remind you of this.

<UEDIT> prompts for the Name, Summary, etc., of the utility you wish to prepare; any fields not prompted for will be calculated automatically. It has extensive "help" facilities.

If *<VTOMFIELDS>* was completed to your satisfaction, you can submit it to the Utility Library via *<USUBMIT>*. Otherwise, you can save the variable (in a workspace or a file) and continue preparing the utility later. To continue preparation, execute the following.

```
      VTOMFIELDS←UEDIT VTOMFIELDS
                .                  ⍝ Specify more fields or
                .                  ⍝ change previously prepared
                .                  ⍝ fields.
RESULT OF <UEDIT> ALSO SAVED IN VARIABLE <UINPUT>.
```

Following are some things to keep in mind when preparing a utility for submission.

* Summary should not assume its readers know, for example, that *<INTEGERS>* is the utility's right argument. The Summary will often be displayed via *<USUMMARY>*, which returns only the title and Summary.

* Description should be reasonably complete. People must be able to decide if the utility does what they want without having to experiment with it.

* Categories should be chosen with special care. If some people will think the utility is in a category that it is not in, they may not be able to find it.

* If the utility is a function, it should contain a comment giving its Utility Library title. The function will likely become part of a system whose maintainers would appreciate knowing where it came from, and where to obtain more extensive documentation on it.

* The utility should not make use of another object that is not in either the library or an I.P. Sharp Associates Program Product. This provides assurance that the utility will not someday fail due to required objects no longer existing.

* If the utility is a function with a subutility that is referenced (mentioned) in one of its comments, then the comment reference should be included in the References field (field 15). For example, if *<ALLOCATE>* is a subutility and *"A SET UP FOR <ALLOCATE>"* is a comment, then the comment should be indexed in the References field. That way, if the subutility is renamed by *<UDEFINE>*, the comment will be updated.

RESULT←UVALIDATE SUBMISSION

SUBMISSION should be a 15-element vector of enclosures--the enclosed fields, usually prepared via *<UEDIT>*. *RESULT* will be either 1 or an error message. If *SUBMISSION* can be successfully submitted to the Utility Library, it will be 1. An error message indicates (only) the first problem that was found with the submission.

For a utility to be successfully submitted, all of its subutilities (except itself, if it is recursive) must first be placed in the library. This presents a problem if, for example, *<UTIL1>* and *<UTIL2>* are subutilities of each other. In such an instance, *<UTIL1>* may first be submitted with its Subutilities field (field 14) empty. After *<UTIL2>* has been submitted, *<UTIL1>* may be resubmitted (via *<URESUBMIT>* or *<UREVISE>*) with a new Subutilities field.

If *SUBMISSION* has been prepared via *<UEDIT>* and all its subutilities are in the library, it is very probable the result of *<UVALIDATE>* will be 1.

MESSAGE←USUBMIT SUBMISSION

This is the function that actually submits utilities to the Utility Library. *SUBMISSION* should be a 15-element vector of enclosures that has been validated by means of *<UVALIDATE>*. The result is a message indicating that the utility has been entered into the library. If *UVALIDATE SUBMISSION* is an error message, executing *'USUBMIT SUBMISSION'* will evoke an error.

MESSAGE←URESUBMIT SUBMISSION

Changes may be made to most of the fields of a utility any time after it has been submitted to the Utility Library. Changes are made via the coversational function *<UREVISE>* or, less commonly, via the non-conversational function *<URESUBMIT>*. Whichever is used, the following fields must remain unchanged: Name, Version, Time, Reviews, Type, and Definition. In certain circumstances, the Utility Librarian will permit the new Definition (field 13) to differ from the old Definition. (Normally, when this is desired, a new version of the utility should be submitted, and the old version deleted.) Requests to have this done should be sent to mailbox code *"ULIB"*.

The right argument of *<URESUBMIT>* should be a 15-element vector of enclosures--the changed submission. The result is a message indicating that the utility has been successfully resubmitted. An error will be evoked if it could not be.

MESSAGE←UREVISE UTILITY

<UREVISE> is a conversational function used for changing an existing utility or submitting a new version of an existing utility. The right argument should specify the utility you wish to change or submit a new version of. The function will prompt you to change the fields. To submit a new version of the utility, change the Version (field 2) to 1 + latest version number.

The result of *<UREVISE>* is a message indicating that the revision was made.

MESSAGE←[DELETED] UDELETE UTILITY

(See the description of the Deleted field (field 10) for details of how deleted utilities are treated.) This function deletes the specified utility from the Utility Library. *<UTILITY>* should specify the utility you wish to delete. *DELETED* should be a character vector detailing why the utility is being deleted from the Utility Library (i.e. the Deleted field (field 10)). The maximum width of *<DELETED>* is 80 characters. If *<DELETED>* is not specified, you will be prompted to enter it.

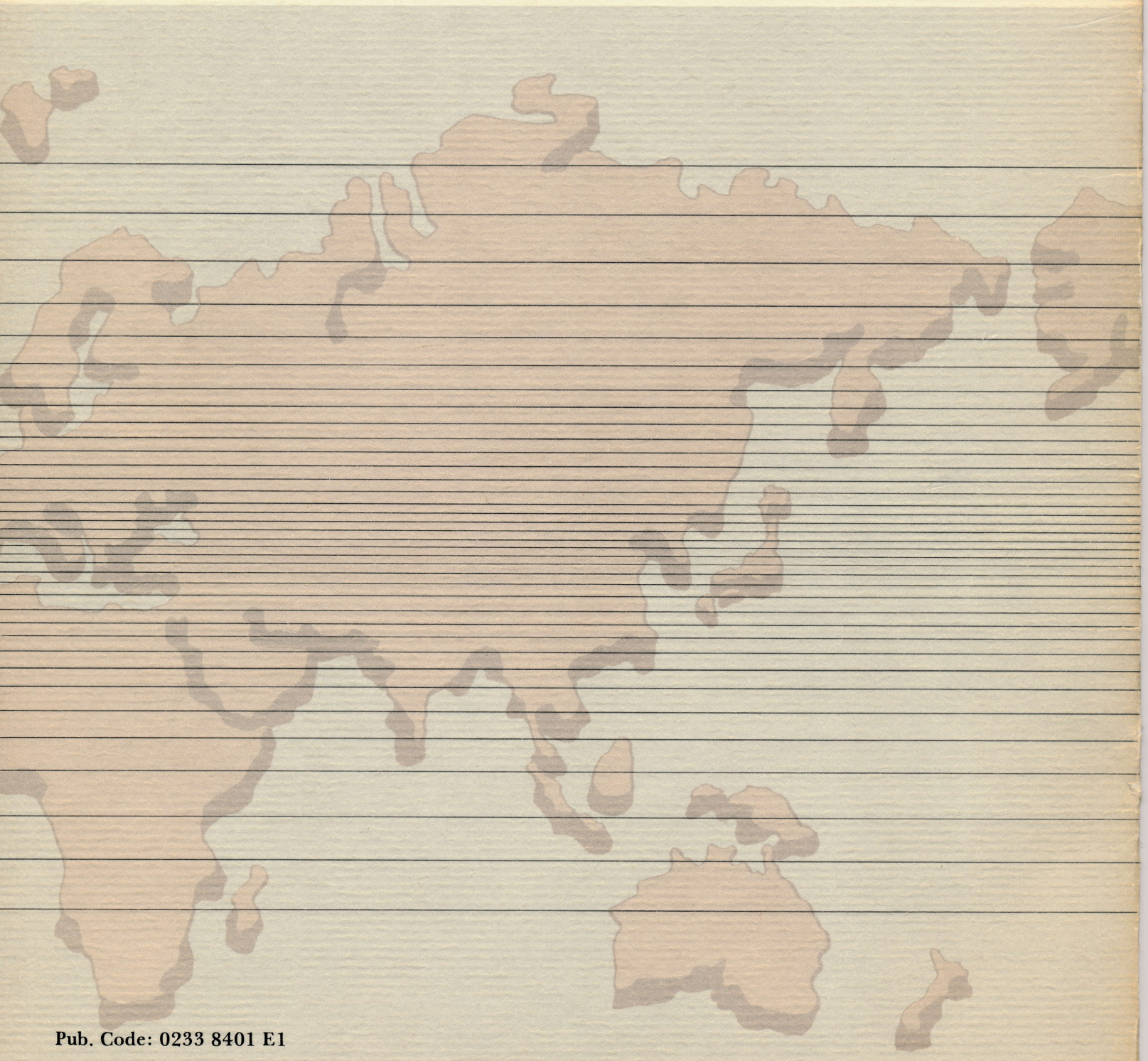
The result of *<UDELETE>* is a message indicating that the utility has been deleted. An error will be evoked if it could not be.

Appendix

THE CATEGORY INDEX

REVISION DATE: 1983-08-27

```
 1  MISCELLANEOUS
 2  STRUCTURAL TRANSFORMATIONS  ⍝ RESHAPING, CATENATING, TRANSPOSING, ETC.
40  SELECTING FROM ARRAYS  ⍝ INDEXING, TAKE, COMPRESSION, ETC.
41  BOOLEAN ARRAYS
 3  PARTITIONED ARRAY HANDLING
 4  ENCLOSED ARRAYS
 5  NUMERIC CALCULATION
42  LARGE ARRAYS
 6  SORTING AND GRADING
 7  SEARCHING
 8  PACKAGES
 9  FILES
43     FILE PRIMITIVE SIMULATION
32     FILE PERMISSION CONTROL
11     CONCURRENT USE OF FILES  ⍝ FILE SHARING
12     FILE ORGANIZATION
13     FILE TOOLS
44  EVENT TRAPPING
14  COMMUNICATION BETWEEN TASKS
15     SHARED VARIABLES
10     MAILBOX
17  S-TASKS
18  N-TASKS
16  B-TASKS
20  TERMINAL INPUT/OUTPUT
21     TERMINAL INPUT
22     TERMINAL CONTROL
23        TABS
45  GRAPHICS AND PLOTTING
24  EDITING
25  FORMATTING
30     REPORT FORMATTING
27  TEXT PROCESSING  ⍝ E.G. SPELLING CHECKERS
28  DEFINED FUNCTIONS  ⍝ UTILITIES DEALING WITH DEFINED FUNCTIONS
29     DEBUGGING
26     EXECUTION CONTROL
19     EXECUTION MONITORING
34  MEASURING TIME AND SPACE REQUIREMENTS
35  WORKSPACE TOOLS  ⍝ E.G. WORKSPACE CROSS REFERENCE
36  USAGE CHARGES AND COSTS
37  TIMES AND DATES
38  BUSINESS AND FINANCE
39  STATISTICS AND PROBABILITY
```



Pub. Code: 0233 8401 E1